UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/661,982	09/12/2003	Cary Lee Bates	ROC920000051.D1	9327

46797 7590 02/04/2009

IBM CORPORATION, INTELLECTUAL PROPERTY LAW
DEPT 917, BLDG. 006-1
3605 HIGHWAY 52 NORTH
ROCHESTER, MN 55901-7829

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
02/04/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/661,982	BATES ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 November 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-13* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-13* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated November 14, 2008, responding to the Office action mailed August 20, 2008 provided in the rejection of claims 1-13, wherein claims 1, 6, and 8 have been amended.

Claims 1-13 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Section of Claim Rejections - 35 U.S.C. 112, second paragraph*.

Claim Rejections – 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

2. Claims 1-13 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

3. **As to claim 1**, the claim recites the limitations "... allowing a user to establish ... a relationship between ... deallocators and ... allocators ..." in lines 5-7, and "... determining whether the relationship is violated ..." in lines 12-13. However, there is no further claim limitations cited on how to determine whether the relationship is violated.

At the best, the preamble portion contains "...dynamic allocation during execution code containing a plurality of memory allocators and a plurality of memory deallocators" in lines 2-3 and other portion of claim limitations "... the relationship is represented by a data structure containing a reference to ... deallocators ... allocators" in lines 8-10; however, the relationship represented by a data structure containing a reference to deallocators and allocators is actually input by the user (providing a computer user interface; and allowing a user to establish, via the computer user interface) and not being recorded during execution of code (emphasis added)

Further, in light of the specification of pending application, it recites "... a method for managing memory available for dynamic allocation during execution of code including ... allocators and ... deallocators ... for each deallocator called to free a memory block, determining an associated allocator responsible for allocating the memory block; recoding association information identifying a relationship between each deallocator called and each associated allocator ..." (paragraph [0013] – it describes "*operational relationship*" from view of execution of code); furthermore, it also recites "... allowing a user to establish a relationship between ... deallocators and ... allocators ... allowing the code to execute, upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violate ..." (paragraph [0013] – it describes "*user-established relationship*" from view of a computer user interface – emphasis added)

Thus, even though there exists the *user-established relationship*, without the *operational relationship* the claim does not have a basis to determine whether the (user-

established) relationship is violated. Also, it is not clear for the purpose for the relationship represented by a data structure containing a reference to deallocators and allocators as recited in the claim.

In the interest of compact prosecution, the claim is only subsequently interpreted as any debugger front-end user interface of capable of detecting memory leaks; and not for the operational relationship as recited in the specification for the purpose of further examination (emphasis added)

4. **As to claims 2-5**, they are rejected as including the deficiency in the independent claim 1.

5. **As to claim 6**, the claim recites the limitations "... establishing a relationship between ... deallocator and ... allocator..." in lines 5-7, and "... determining whether the memory space was allocated by the user-selected allocator ... the relationship is violated ..." in lines 10-12. However, there is no further claim limitations cited on how to determine whether the relationship is violated. At the best, the preamble portion contains "...dynamic allocation during execution code containing a plurality of memory allocators and a plurality of memory deallocators" in lines 2-3 and other portion of claim limitations "... the relationship is represented by a data structure containing a reference to ... deallocators ... allocators" in lines 8-10; however, the relationship represented by a data structure containing a reference to deallocators and allocators is actually input by the user (providing a computer user interface; and allowing a user to establish, via the

computer user interface) and not being recorded during execution of code (emphasis added)

Further, in light of the specification of pending application, it recites "... a method for managing memory available for dynamic allocation during execution of code including ... allocators and ... deallocators ... for each deallocator called to free a memory block, determining an associated allocator responsible for allocating the memory block; recoding association information identifying a relationship between each deallocator called and each associated allocator ..." (paragraph [0013] – it describes "*operational relationship*" from view of execution of code); furthermore, it also recites "... allowing a user to establish a relationship between ... deallocators and ... allocators ... allowing the code to execute, upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violate ..." (paragraph [0013] – it describes "*user-established relationship*" from view of a computer user interface – emphasis added)

Thus, even though there exists the *user-established relationship,* without the *operational relationship* the claim does not have a basis to determine whether the (user-established) relationship is violated. Also, it is not clear for the purpose for the relationship represented by a data structure containing a reference to deallocators and allocators as recited in the claim.

In the interest of compact prosecution, the claim is only subsequently interpreted as any debugger front-end user interface of capable of detecting memory leaks; and not

for the operational relationship as recited in the specification for the purpose of further examination (emphasis added)

6. **As to claim 7**, it is rejected as including the deficiency in the independent claim 6.

7. **As to claim 8**, the claim recites the limitations "... setting an upper limit on the amount of memory space an allocator can allocate ..." in lines 4-5, and "... when the amount of memory space allocated exceeds the limit ..." in line 10. However, there is no further claim limitations cited on how to determine when the amount of memory space allocated exceeds the limit. At the best, the preamble portion contains "...dynamic allocation during execution code containing a plurality of memory allocators and a plurality of memory deallocators" in lines 1-3 and other portion of claim limitations "... the upper limit and a reference to the allocator are stored in a data structure ..." in lines 6-7; however, the upper limit and a reference to the allocator stored in a data structure is actually input by the user (providing a computer user interface; and allowing a user to establish, via the computer user interface) and not being recorded during execution of code (emphasis added)

Further, in light of the specification of pending application, it recites "... a method for managing memory available for dynamic allocation during execution of code including ... allocators and ... deallocators ... for each deallocator called to free a memory block, determining an associated allocator responsible for allocating the

memory block; recoding association information identifying a relationship between each deallocator called and each associated allocator ..." (paragraph [0013] – it describes "*operational relationship*" from view of execution of code); furthermore, it also recites "... allowing a user to establish a relationship between ... deallocators and ... allocators ... allowing the code to execute, upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violate ..." (paragraph [0013] – it describes "*user-established relationship*" from view of a computer user interface – emphasis added)

Thus, even though there exists the *user-established relationship,* without the *operational relationship* the claim does not have a basis to determine whether the (user-established) upper limit is violated. Also, it is not clear for the purpose for the upper limit and a reference represented by a data structure as recited in the claim.

In the interest of compact prosecution, the claim is only subsequently interpreted as any debugger front-end user interface of capable of detecting memory leaks; and not for the operational relationship as recited in the specification for the purpose of further examination (emphasis added)

8. **As to claims 9-13**, they are rejected as including the deficiency in the independent claim 8.

Claim Rejections – 35 USC § 102(e)

The following is quotation of 35 U.S.C. 102(e) which form the basis for all obviousness rejections set forth in this office action:

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

9. Claims 1-8 and 11-13 are rejected under 35 U.S.C. 102(e) as being anticipated by Spertus et al. (Pat. No. 6,938,245 B1) (hereinafter 'Spertus')

10. **As to claim 1** (Currently Amended), Spertus discloses a computer-implemented method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, comprising:

- Providing a computer user interface (e.g., Fig. 1, elements 111(a) – UI Client; 125 – User Input; 127 – Formatted Output; Fig. 2, element 251 – CLI Client; Figs. 4-8; Col. 2, Lines 33-37 - ... the interactive interface can be used not only with debug information from a current execution of the program …);
- allowing a user to establish, via the computer user interface, a relationship between one or more of the memory deallocators and one or more of the memory allocators, wherein the relationship requires that memory space allocated by the one or more allocators is freed by the one or more deallocators,

and wherein the relationship is represented by a data structure containing a reference to the one or more of the memory deallocators and the one or more of the memory allocators (e.g., Fig. 9; Col. 15, Lines 20-42 – Fig. 9 shows the structure of a bucket 901. Each bucket 901 is made up of a page **information structure** 907 which includes information 904 indicating the size of **the objects that will be allocated** from the bucket and **a free list pointer** 906 **indicating the head of the list of unallocated object** 911 contained in bucket 901 …);

- allowing the code to execute; upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violated (e.g., Fig. 8, element 811 – *gcLogAllLeaks* - … write all leaks to the log; Col. 2, Lines 59-62 - … provides memory debugging information such as memory allocations, **memory leaks**, and current heap size; Col. 5, Lines 51-67 - … One problem detected by a memory debugger is memory 'leaks' …); and
- if so, notifying the user (e.g., Fig. 1; Col. 5, Lines 48-52 – Debugger client 101 and the user interface clients 111 further communicate with each other by means of control channel 121, which may be any arrangement which permits transfer of messages between debugger client 102 and a user interface client 111)

11. **As to claim 2** (Original) (incorporating the rejection in claim 1), Spertus discloses the method wherein notifying the user comprises halting execution of the code (e.g., Col. 4, Lines 56-65 - … may instruct the debugger to stop execution of program …)

12. **As to claim 3** (Original) (incorporating the rejection in claim 1), Spertus discloses the method wherein notifying the user comprises halting execution of the code and displaying a status message to the user (e.g., Col. 4, Lines 56-65 - ... may instruct the debugger to stop execution of program ...; Fig. 1; Col. 5, Lines 48-52 – Debugger client 101 and the user interface clients 111 further communicate with each other by means of control channel 121, which may be any arrangement which permits transfer of messages between debugger client 102 and a user interface client 111)

13. **As to claim 4** (Original) (incorporating the rejection in claim 1), Spertus discloses the method if the relationship is not violated, freeing the memory space (e.g., Fig. 8, element 805 – *gcEnableFree* - ... cause explicit calls to free() and delete to reclaim memory; Fig. 10; Col. 6, Lines 7-10 - ... automatic memory management by periodically collecting garbage, that is, memory which was once allocated but is not longer being used by the program, and freeing the garbage memory for reuse)

14. **As to claim 5** (Original) (Original) (incorporating the rejection in claim 1), Spertus discloses the method wherein determining whether the relationship is violated comprises determining that the memory space was allocated by an allocator different from the one or more memory allocators (e.g., Fig. 8, element 811 – *gcLogAllLeaks* - ... write all leaks to the log; Col. 2, Lines 59-62 - ... provides memory debugging information such as memory allocations, **memory leaks**, and current heap size; Col. 5, Lines 51-67 - ... One problem detected by a memory debugger is memory 'leaks' ...)

15. **As to claim 6** (Currently Amended), Spertus discloses a computer-implemented method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, comprising:

- establishing a relationship between a user-selected memory deallocator and a user-selected memory allocator, wherein the relationship requires that memory space freed by the user-selected deallocator have been allocated by the user-selected allocator, and wherein the relationship is represented by a data structure containing a reference to the user-selected deallocator and the user-selected allocator (e.g., Fig. 9; Col. 15, Lines 20-42 – Fig. 9 shows the structure of a bucket 901. Each bucket 901 is made up of a page **information structure** 907 which includes information 904 indicating the size of **the objects that will be allocated** from the bucket and **a free list pointer** 906 **indicating the head of the list of unallocated object** 911 contained in bucket 901 ...);
- allowing the code to execute (e.g., Col. 1, Lines 17-18 – Debuggers are tools used by programmers to determine what is going on when a program is executed);
- upon a call to the user-selected deallocator to free a memory space, determining whether the memory space was allocated by the user-selected allocator (e.g., Fig. 8, element 811 – *gcLogAllLeaks* - ... write all leaks to the log; Col. 2, Lines 59-62 - ... provides memory debugging information such as memory allocations,

memory leaks, and current heap size; Col. 5, Lines 51-67 - … One problem detected by a memory debugger is memory 'leaks' …); and

- if so, notifying the user that the relationship is violated (e.g., Fig. 1; Col. 5, Lines 48-52 – Debugger client 101 and the user interface clients 111 further communicate with each other by means of control channel 121, which may be any arrangement which permits transfer of messages between debugger client 102 and a user interface client 111)

16. **As to claim 7** (Original) (incorporating the rejection in claim 6), please refer to claim **3** as set forth above accordingly.

17. **As to claim 8** (Original), Spertus discloses a method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, comprising:

- setting an upper limit on the amount of memory space an allocator can allocate during execution of the code, wherein the upper limit is specific to the allocator; wherein the upper limit and reference to the allocator are stored in a data structure, thereby relating the upper limit to the allocator (e.g., Col. 17, Lines 15-25 - At initialization time, the allocator creates arena 0116 by mapping a large sequence of logical pages 905. This mapping operation **reserves** the virtual address space acquired for the logical pages);

- during execution of the code, tracking the amount of memory space allocated by the allocator (e.g., Col. 17, Lines 26-46 - ... During a request for allocation of a large object ... Finally, the allocation request is satisfied ...); and
- when the amount of memory space allocated exceeds the limit, notifying a user (e.g., Fig. 1; Col. 5, Lines 48-52 – Debugger client 101 and the user interface clients 111 further communicate with each other by means of control channel 121, which may be any arrangement which permits transfer of messages between debugger client 102 and a user interface client 111)

18. **As to claim 11** (Original) (incorporating the rejection in claim 8), please refer to claim **7** as set forth above accordingly.

19. **As to claim 12** (Original) (incorporating the rejection in claim 8, Spertus discloses wherein the upper limit is independent of other memory size limitations (e.g., Fig. 9; Col. 15, Lines 20-42 – Fig. 9 shows the structure of a bucket 901. Each bucket 901 is made up of a page **information structure** 907 which includes information 904 indicating the size of **the objects that will be allocated** from the bucket and **a free list pointer** 906 **indicating the head of the list of unallocated object** 911 contained in bucket 901 ...)

20. **As to claim 13** (Original) (incorporating the rejection in claim 8), Spertus discloses wherein the upper limit is not a limit on a stack size (e.g., Col. 17, Lines 44-46

- ... If the number of consecutive uncommitted pages in arena 1006 is not enough, another large group of uncommitted memory is mapped and added to arena 1006)

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

21. Claims 9-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Spertus in view of Kolawa et al. (Pat. No. 5,842,019) (hereinafter 'Kolawa')

22. **As to claim 9** (Original) (incorporating the rejection in claim 8), Spertus discloses a debugger which may be easily adapted to a number of different kinds of user interfaces, including the user interface provided by Web browsers and that works as well to analyze information about past executions of a program as it does to analyze information about a current execution (e.g., Col. 2, Lines 11-19) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Method and System for Dynamically Detecting Leaked Memory Space in a Computer Program*, Kolawa discloses the step of tracking comprises:

- determining whether the allocator is called to allocate memory and, if so, incrementing a counter (e.g., Fig. 8, blocks 80 – NEW BLOCK?; 81 – INCREMENT REFERENCE COUNT; Col. 6, Lines 1-7 - … points to a newly allocated memory block (block 80), the reference count is incremented by one (block 81) …); and
- determining whether a deallocator is called to deallocate memory allocated by the allocator and, if so, decrementing the counter (e.g., Fig. 8, blocks 82 – OLD BLOCK?; 83 – DECREMENT REFERENCE COUNT; Col. 6, Lines 1-7 - … if a new value has been assigned to the pointer variable, thereby eliminating the reference to the old memory block (block 82), the reference count is decremented by one (block 83) …)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kolawa into the Spertus' system to further provide other limitations stated above in the Spertus system.

The motivation is that it would further enhance the Spertus' system by taking, advancing and/or incorporating the Kolawa's system which offers significant advantages of an approach to detecting leaked memory space in a computer program by employing a combination of runtime, dynamic memory searching and post-execution memory sweeping as once suggested by Kolawa (e.g., Col. 1, Lines 41-46)

23. **As to claim 10** (Original) (incorporating the rejection in claim 8), Kolawa discloses the step of tracking comprises incrementing a counter in the event of memory

allocation by the allocator (e.g., Fig. 8, blocks 80 – NEW BLOCK?; 81 – INCREMENT REFERENCE COUNT; Col. 6, Lines 1-7 - ... points to a newly allocated memory block (block 80), the reference count is incremented by one (block 81) ...) and decrementing the counter in the event of memory deallocation of memory space allocated by the allocator (e.g., Fig. 8, blocks 82 – OLD BLOCK?; 83 – DECREMENT REFERENCE COUNT; Col. 6, Lines 1-7 - ... if a new value has been assigned to the pointer variable, thereby eliminating the reference to the old memory block (block 82), the reference count is decremented by one (block 83) ...)

Conclusion

24. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Ben C. Wang
Examiner, Art Unit 2192

/Eric B. Kiss/
Eric B. Kiss
Primary Examiner, Art Unit 2192